General Autodiscovery of DTN Nodes

Jim Wyllie

Verizon / NASA / Ohio University

April 16, 2007

Advisors: Wesley Eddy, Shawn Ostermann, Joe Ishac

Quick DTN Overview

- Delay / Disruption Tolerant Networking
- Potentially high RTT (seconds, minutes, hours)
- Not always a direct end-to-end path (maybe never)
- Potentially high error rates
- Deep space, sensor networks, etc
- Directly below application layer

Current DTN Autodiscovery Solutions

Generally lumped into three categories:

- Ignored
  - Assumes hosts are up (error if not)
  - Manual configuration of contact times
  - Typical for research / testing environments
- Home-brew
  - Added in a domain- and implementation-specific way
  - Typically has all the limitations of an ad-hoc implementation
- Punted
  - Assumes the lower layer can queue / figure it out
  - Can lead to bad link utilization decisions

Problems with current solutions

- No standardization on mechanisms
- Typically only supports certain underlying layers
- Security is typically non-existent
- Interoperability is virtually non-existent

Why this is hard

- Long vs short delays
- Ample vs minimal power
- Highly disruptive vs "long" uptimes

How do we generalize all of these into one protocol?

Two-layered approach

- Look at the problem in two pieces
  - Generic to all autodiscovery systems
  - Domain-specific implementation

Generic Layer

- Consists of a new block type
- Has the following fields:
  - **Autodiscovery Flags** — Flags about autodiscovery, contained fields
  - **Autodiscovery Protocol** — Flags about the type of autodiscovery payload
  - **Bundle node ID** — Reference to the node ID to which this autodiscovery refers
  - **Start Time** — Start time of contact (omit for "right now")
  - **End time** — End time of contact (omit for "in perpetuity")

Timestamps have seconds-based granularity

Autodiscovery Flags

- Bit 1 — Bundle contains contact time information
- Bit 2 — Bundle contains start time[1]
- Bit 3 — Bundle contains end time[1]
- Bits 4-7 are reserved

[1] Must be clear if bit 1 is clear

Autodiscovery Protocol Field

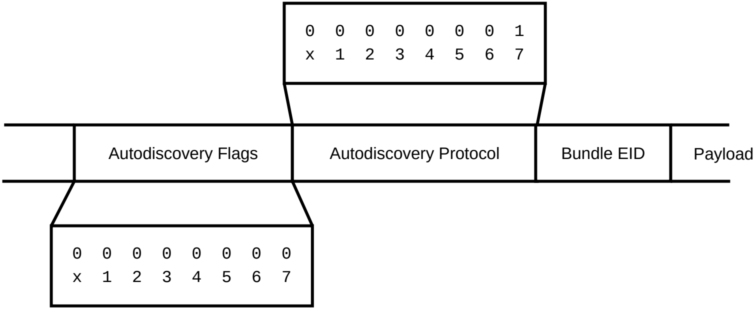
- Determines the type of domain-specific autodiscovery data
- Gives structure to the payload data
  - 0x00 is unused
  - 0x01 defines a NASA discovery protocol
- Not present if connection times are present

How these parts work together

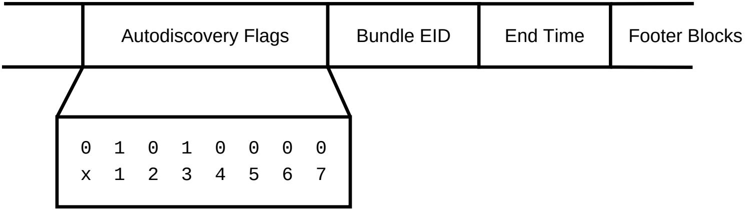
- Last packet in the sequence: contact time is set
- If flag is clear, flags / protocol / times must not be present
- Assuming N packets in entire Autodiscovery sequence:
  - $[0, N-1]$ — No contact time: domain-specific discovery payload
  - N — Contact time set: start and end contact times from discovery

Sample packet: domain-specific discovery

These packets aren’t always needed: depends on the deployment environment



Sample packet: connection times known



Autodiscovery Flags

Bundle EID

End Time

Footer Blocks

0	1	0	1	0	0	0	0
x	1	2	3	4	5	6	7

Advantages to such an approach

- "Separate but compatible" autodiscovery mechanisms for DTN
- Can provide standardized security against autodiscovery attacks (DoS, MITM)
- Allows domain-specific autodiscovery mechanisms
- Lower-layer independent

Potential security concerns

- Deep-space scenario
  - Rover spots aliens. Foreign orbiter discovered
  - Satellite discovers hacker’s dish
  - X.509 Certificates? Pre-shared keys?
- Public-use scenario
  - On a bus, discover a rogue PDA as router
  - Key fingerprints a la SSH?

Potential security concerns

- 3rd party reports a host is down imminently
- Send early, send often
- Mitigated by "trusted" DTN network (X.509, PSKs)

Summary

- Standardized system for DTN autodiscovery
- General and optional domain-specific portion
- Usable across a wide variety of domains
- Discovery can be abstracted out of implementation
- Can be secured if necessary

Questions?

- Any questions?